Utilisation de la densité des données dans l'apprentissage des réseaux RBF

Eric Bertrand Fokou Dzokou, Paulin Melatagia Yonta, Narcisse Talla Tankam

Utilisation de la densité des données dans l'apprentissage des réseaux RBF

Eric B. Fokou Dzokou[1] — Paulin Melatagia Yonta[1,2] — Narcisse Talla Tankam [3,4]

[1] LIRIMA, Equipe IDASCO, Faculté des Sciences, Département d'Informatique, Université de Yaoundé I B.P. 812 Yaoundé, Cameroun.

[2] UMI 209 UMMISCO, Université de Yaoundé I, B.P. 337 Yaoundé, Cameroun.

[3] Laboratoire Automatique et Informatique Appliquée (LAIA), IUT Fotso Victor de Bandjoun, Université de Dschang, Cameroun.

[4] Laboratoire d'Electronique et du Traitement du Signal (LETS), Ecole Nationale Supérieure Polytechnique, Yaoundé, Cameroun.

E.mail : fokoub@gmail.com, paulinyonta@gmail.com, narcisse.talla@gmail.com

RÉSUMÉ. Dans cet article nous abordons le problème d'apprentissage avec les réseaux dit à fonction de base radiale (RBF). Nous proposons un nouvel algorithme de construction des réseaux RBF flou pour la classification. Les caractéristiques principales de cet algorithme sont sa stabilité par rapport à l'algorithme classique des réseaux RBF (qui utilise k-moyennes sur la couche cachée) et sa précision par rapport à l'algorithme de Keramitsoglu et al. [1] dont il est issu. Le nouvel algorithme d'apprentissage calcule automatiquement le nombre de neurones sur la couche cachée et les paramètres des neurones RBF en exploitant les zones de forte densité. Les expérimentations effectuées en reconnaissance de formes fournissent de meilleurs résultats que ceux de Keramitsoglu et al. avec un court temps d'exécution.

ABSTRACT. In this paper we discuss the learning problem of Radial Basis Function (RBF) Neural Networks. We propose a new learning algorithm of fuzzy RBF networks for classification. The main features of this algorithm are its stability compared to the classical learning algorithm of RBF networks (which uses k-means on the hidden layer) and accuracy with respect to the algorithm of Keramitsoglu et al. [1] from which it is derived. The new learning algorithm automatically calculates the number of neurons on the hidden layer and the parameters of the RBF neurons by exploiting high density areas. The experimentations carried out in pattern recognition provide better results than those of Keramitsoglu et al. with a shorter running time.

MOTS-CLÉS : Apprentissage automatique, Réseaux de neurones, Réseaux RBF, Fuzzy RBF.

KEYWORDS : Machine learning, Neural networks, RBF networks, Fuzzy RBF.

1. Introduction

Dès la fin des années 80, de nombreux modèles neuronaux ont émergés ; parmi ces modèles le perceptron multicouche [2] a été le plus populaire. Cependant cet algorithme est couteux en temps dû à son architecture multicouches et aux propriétés de l'algorithme de rétro-propagation du gradient, tout ceci fortement accentué par le volume important de données à traiter de nos jours. Les réseaux RBF [3] présentent alors des avantages incluant une architecture simple avec une unique couche cachée, et un algorithme d'entrainement plus simple. La principale difficulté dans la construction des réseaux RBF concerne la question du nombre de neurones radiaux à utiliser pour une application donnée et cette architecture souffre du problème de l'augmentation exponentielle du nombre de neurones cachés requis en fonction de la dimension de l'espace d'entrée.

Nous proposons dans ce papier un nouvel algorithme qui utilise les moyennes floues [4] pour construire des réseaux RBF. L'algorithme utilise les techniques de discrétisation [5] pour déterminer le nombre de neurones radiaux mais aussi leur position sur chaque axe de l'espace d'entrée. Cette approche permet aux centres de ces neurones radiaux d'être placés régulièrement par rapport à la distribution des données. Afin de mettre sur pied un algorithme le plus stable possible, nous ajoutons un paramètre de régularisation [6] aux poids des neurones sur la couche de sortie. À partir des observations effectuées : temps d'exécution, taux d'apprentissage, stabilité, nous déduisons l'importance des modifications apportées pour la construction du réseau. Nous obtenons par expérimentation de bons résultats de généralisation sur des jeux de données bien reconnus en apprentissage automatique par rapport à l'algorithme de Keramitsoglu et al.

La suite de l'article est organisée comme suit : la section 2 présente les algorithmes d'apprentissage des RBF ; la section 3 décrit l'algorithme d'entrainement des réseaux que nous proposons. La section 4 présente les résultats et des discussions sur les expérimentations effectuées. Nous terminons par une conclusion à la section 5.

2. Algorithmes d'apprentissage des réseaux RBF

Un réseau RBF est constitué de deux couches principales : une couche cachée ou radiale et une couche de sortie. La fonction de transfert sur la couche cachée est une fonction de base radiale et sur la couche de sortie on dispose des neurones linéaires ou sigmoïdes selon le problème. Plusieurs fonctions radiales peuvent être utilisées [7], mais la plus courante est une fonction ϕ de type gaussienne :

$$\phi_i(x) = exp\Big(- \frac{r^2}{2\sigma_i^2} \Big) = \exp\left(- \frac{\| x - w_i \|^2}{2\sigma_i^2} \right) \tag{1}$$

où $\| \ . \ \|$ représente la norme euclidienne, w_i désigne la position (le centre) du neurone radial i dans son espace d'entrée et σ_i désigne la largeur de la fonction gaussienne : $\phi_i(x)$ est donc la sortie du neurone radial i.

Le réseau approxime une fonction f avec l'expression suivante :

$$\hat{f}(p) = W^2 \Phi(p) \tag{2}$$

où $\Phi = \left[\phi_1, \phi_2 \dots \phi_{S^1}\right]^T$ représente la sortie des neurones cachées, S^1 le nombre de neurones sur la couche cachée et W^2 est la matrice des poids sur la couche de sortie ;

$\phi_i(x)$ est la sortie du neurone caché de centre w_i et σ_i est sa variance. L'entrainement d'un réseau RBF s'effectue en deux étapes distinctes : la première consistant à estimer la position des centres des neurones radiaux puis leur variance, et la deuxième à estimer les poids des neurones sur la couche de sortie.

La première alternative pour positionner les centres des neurones radiaux est de les fixer sur certains exemples du jeu d'apprentissage p_k choisis aléatoirement ; le calcul des variances des neurones radiaux est fait par la formule $\sigma = \frac{d}{\sqrt{2M}}$ où d est la distance maximale entre 2 neurones RBF et M est le nombre de neurones RBF. Une deuxième alternative consiste à positionner les centres des neurones radiaux à l'aide de l'une des méthodes d'apprentissage non supervisé, k-moyennes [8] ou cartes auto-organisatrice de Kohonen [9]. Le calcul des variances peut s'effectuer alors de la manière suivante [10] :

$$\sigma_i^2 = \frac{1}{N} \sum_{j=1}^{N} \parallel p_j - w_i \parallel^2 \tag{3}$$

où w_i est le centre du cluster i et $\{p_1, p_2,\ldots,p_N\}$ est l'ensemble des N plus proches exemples du centre w_i.

Pour l'estimation des poids de la couche linéaire, l'algorithme d'entrainement du perceptron simple (règle du Least Mean Square) [11] :

$$W^2(t+1) = W^2(t) + e(t)\Phi^T \tag{4}$$

avec $e(t) = d(t) - a(t)$ qui représente le vecteur mesurant l'erreur entre les sorties cibles et les sorties calculées du réseau au temps t. Cette estimation peut également être faite en utilisant $W^2 = DP^+$, où $D = \left[d_1, d_2 \ldots d_Q\right]$ représente la matrice des réponses désirées du réseau et P^+ la matrice pseudo-inverse de Moore-Penrose [12].

La partie la plus couteuse en temps dans l'entrainement des réseaux RBF est le positionnement des neurones cachés avec l'algorithme k-moyennes qui est la méthode communément utilisée. Cependant cette méthode nécessite plusieurs passages sur tous les patterns d'entrainement. Il faut également souligner que l'algorithme doit être exécuté plusieurs fois, puisque le nombre de neurones cachés idéal est sélectionné par essai et erreur. Un inconvénient plus important encore est le fait que l'algorithme standard manque de stabilité, étant donné que les résultats dépendent de la première sélection aléatoire des centres sur la couche cachée.

Keramitsoglu et al. [1], dans leur article apportent des débuts de réponse à certaines de ces limites. Ils proposent un algorithme qui utilise les moyennes floues [4] pour positionner les centres des neurones radiaux. Cet algorithme permet non seulement de déterminer le nombre de neurones radiaux, mais aussi leur position et leur variance. L'algorithme commence par partitionner l'univers de départ de chaque variable en entrée x_i $(i = 1, 2, \ldots, R)$ en c_i ensembles flous triangulaires[1] $A_i^1, A_i^2, \ldots, A_i^{c_i}$. Une fonction membre est associée à chaque ensemble flou A_i et permet de calculer la valeur d'adhésion d'une coordonnée p_i d'un pattern p. Cette fonction est définie comme suit :

$$\mu_{A_i^j}(p_i) = \begin{cases} 1 - \dfrac{|p_i - a_i^j|}{\delta}, & \text{si } p_i \in [a_i^j - \delta, a_i^j + \delta] \\ 0, & \text{sinon} \end{cases} \tag{5}$$

1. Le terme triangulaire provient de l'allure de la fonction valeur absolue

où a_i^j est le centre de l'intervalle d'adhésion défini par la fonction $\mu_{A_i^j}$ et δ est la moitié de la largeur de l'intervalle. Les centres des ensembles flous constituent alors une grille multidimensionnelle sur l'espace d'entrée et les nœuds de la grille représentent l'ensemble des candidats pour devenir les centres des neurones cachés. L'algorithme 1 permet de sélectionner le nœud le plus approprié pour un exemple en entrée. L'algorithme des moyennes floues [1] évalue alors la distance entre chaque nouvel exemple et les sous-espaces flous déjà générés ; si la plus petite de cette distance est supérieur à 1, un nouveau sous-espace flou est généré. A la fin de l'exécution, l'ensemble des sous-espaces flous constitue alors les centres des neurones cachés.

Algorithme 1: Génération du plus proche sous-espace flou associé a un exemple

Entrées :
$x = [x_1, x_2, \ldots, x_R]^T$, pattern en entrée.
$A_i = \{A_i^1, A_i^2, \ldots, A_i^{c_i}\}$, ensemble des ensembles flous pour la variable x_i.
Sorties :
A, plus proche sous-espace flou associé à x.

1. Pour $i = 1, 2, \ldots, R$ sélectionner l'ensemble flou A_i^k qui assigne la valeur maximale d'adhésion à x_i.
2. Générer le sous-espace flou A en combinant les ensembles flous sélectionnés à l'étape 1.

Cependant l'algorithme de Keramitsoglou possède également des limites. Le premier étant le nombre c_i d'ensembles flous triangulaires fixé également par essai et erreur et deuxièmement, la méthode de positionnement de manière équidistante des centres des ensembles flous sans tenir compte de la distribution des exemples en entrée constituent également une limite. Dans cet article nous allons donc proposer un nouvel algorithme d'entrainement qui se base sur les travaux de Keramitsoglou et al., et qui corrige les deux problèmes ci-dessus identifiés.

3. Algorithme proposé

Nous proposons un nouvel algorithme pour les réseaux RBF, stable, avec de bon taux d'apprentissage mais aussi qui réduit le nombre de paramètres à fixer par un utilisateur. En effet, comme nous l'avons déjà mentionné, l'algorithme de Keramitsoglou et al. [1] bien que possédant de bonnes propriétés est limité par le choix fait par l'utilisateur du nombre d'ensembles flous à fixer. En tenant compte de la diversité des jeux de données ce nombre peut être très différent d'un cas à l'autre ; pour cela en procédant par essai et erreur, le temps requis pour ajuster correctement ce paramètre peut être important. De plus cet algorithme positionne les centres des ensembles flous de manière équidistant ; ceci implique alors que l'algorithme suppose une distribution uniforme des exemples dans l'espace, ce qui est rarement le cas dans la pratique.

Notre solution consiste alors à déterminer de manière automatique le nombre adéquat de neurones radiaux mais aussi leur position afin de couvrir efficacement l'espace. Pour ce faire nous utilisons une technique bien connue en fouille de données et en apprentissage automatique : la *discrétisation des attributs continus* [5]. La discrétisation est une étape importante dans certains traitements [13], où les attributs continus sont transfor-

més en valeurs discrètes. En effet discrétiser des données consiste à convertir les valeurs continues en un petit nombre d'intervalles définis par des points de coupe et attribuer ensuite à chaque valeur continue qui se situe dans un intervalle, une valeur discrète. Des méthodes de discrétisation ont été proposées dans la littérature [14, 15, 16], cependant le problème de déterminer la coupe optimale est NP-difficile [17]. Les méthodes de discrétisation se basent sur des critères statistiques, informelles, ou encore d'autres critères dédiés. Ces méthodes peuvent être regroupées en classes [18] : supervisé/non-supervisé, globale/locale, statique/dynamique. Nous nous intéressons aux méthodes supervisées et non-supervisées, la différence provenant du fait que l'information de classe est utilisée ou non. Parmi les méthodes non-supervisées nous avons la discrétisation à égale largeur et celle à égale fréquence [5]. La première détermine la valeur minimale et maximale de l'attribut à discrétiser et divise la plage ainsi définie en un nombre fixé par l'utilisateur d'intervalles de même largeur. La seconde divise la plage en intervalle de telle sorte que chaque intervalle contient approximativement le même nombre de données. Un de nos objectifs étant de réduire le nombre de paramètres à fixer par l'utilisateur, nous devons donc trouver une technique pour déterminer automatiquement le nombre d'intervalles à considérer. Afin de déterminer ce nombre, nous utilisons des algorithmes qui permettent de calculer le nombre adéquat d'intervalles à utiliser dans la construction d'un histogramme [19, 20, 21]. Ces techniques bien que anciennes restent encore aujourd'hui largement utilisées dans la plupart des travaux. Sturges [19] propose un algorithme qui se base essentiellement sur le nombre de données en entrée. Scott [20] propose une méthode basée sur l'écart type et le nombre de données en entrée et Freedman et Diaconis [21] dans leurs travaux proposent une méthode qui se base sur l'intervalle inter-quartile. Les techniques de discrétisation supervisées [22, 14, 15, 16] ont fait également leurs preuves ; parmi les méthodes supervisées récentes et communement utilisées, nous avons utilisé la discrétisation basée sur le coefficient de contingence de l'attribut de classe (CACC) [14] et celle basée sur la maximisation de l'interdépendance de l'attribut de classe (CAIM) [15].

Afin de capturer plus de corrélation entre les attributs, nous utilisons un modèle gaussien multivarié [23] sur les neurones de la couche cachée défini comme suit :

$$\phi_i(x) = \exp\left(- \frac{(x - w_i)^T \Sigma^{-1} (x - w_i)}{2} \right) \tag{6}$$

où Σ représente la matrice de covariance. Ce modèle est bien adapté pour les réseaux RBF, car ces réseaux sont communément utilisés sur des données dont le nombre d'attributs n'est pas important, réduisant ainsi le coût de calcul de la matrice inverse Σ^{-1}.

La stabilité de notre algorithme algorithme provient de l'élimination de l'algorithme des k-moyennes dont les résultats dépendent de la première sélection aléatoire des centres sur la couche cachée. Ce choix aléatoire peut conduire dans certains cas à un résultat peu satisfaisant et donc instable. L'algorithme proposé qui ne fait pas de choix aléatoire permet quelque soit le nombre d'exécution de générer des centres sur la couche cachée qui couvrent adéquatement l'espace en entrée. Pour améliorer encore la stabilité de notre algorithme, nous modifions la formule de mis-à-jour des poids des neurones sur la couches de sortie. On ajoute alors un paramètre de régularisation [6] afin de réduire l'amplitude des poids des neurones, permettant ainsi de résoudre le problème de surapprentissage. La formule de mise a jour des poids dans le cas où nous considérons des neurones linéaires devient alors :

$$w_{i,j}^{t+1}(x) = w_{i,j}^{t}(x) - \eta \left((d_i - a_i)x + \frac{\lambda}{m} w_{i,j}^{t}(x) \right) \tag{10}$$

Algorithme 2: Apprentissage RBF avec densité des données

Entrées :

$X = \{x_1, x_2, \ldots, x_Q\}$, ensemble de Q patterns en entrée.

Sorties :

Taille et centres de la couche cachée.

1. Normaliser X.
2. Discrétiser X. Choisir un des algorithmes suivant :
   - cas non-supervisé : Sturges, Scott, Freedman et Diaconis.
   - cas supervisé : CACC, CAIM.
3. Définir les points de coupe comme les centres des ensembles flous.
4. Initialiser le nombre de sous-espace flou, $L = 0$.
5. Considérer le premier exemple en entrée x_1 et générer le premier sous-espace flou $A^1 = \{\alpha^1, \delta^1\}$. Modifier $L = 1$.
6. Supposer que $k-1$ exemples ont été examinés et L sous-espace flous ont été générés avec $1 \leq L \leq K-1$. Insérer le k^{ieme} pattern x_k et calculer la distance $rd^l(x_k)$ $(l = 1, 2, \ldots, L)$ entre x_k et chacun des L sous-espaces floues à partir de l'équation :

$$rd^l(x_k) = \begin{cases} \dfrac{\parallel A^l - x_k \parallel}{\parallel \delta^l \parallel}, & \text{si } \parallel A^l - x_k \parallel \leq \parallel \delta^l \parallel \\ 1, & \text{sinon.} \end{cases} \tag{7}$$

La plus petite distance $rd^{l_0}(x_k)$ correspond au sous-espace flou $A^{l_0} = \{\alpha^{l_0}, \delta^{l_0}\}$. Si l'inégalité $rd^{l_0}(x_k) < 1$ est vérifiée aller à l'étape 7 sinon aller à l'étape 6.

7. Générer un nouveau sous-espace flou pour l'exemple x_k avec l'Algorithme 1. Modifier $L = L + 1$.
8. Si $k = Q$ stop sinon insérer un nouvel exemple et aller à l'étape 6.
9. Calculer les sorties des neurones de la couche cachée à partir de l'équation :

$$\phi_i(x) = \exp\left(- \frac{(x - w_i)^T \Sigma^{-1} (x - w_i)}{2} \right) \tag{8}$$

.

10. Mise a jour des poids des neurones en sortie à partir de l'équation :

$$w_{i,j}^{t+1}(x) = w_{i,j}^t(x) - \eta\left((d_i - a_i)x + \frac{\lambda}{m} w_{i,j}^t(x) \right) \tag{9}$$

où $w_{i,j}^t$ représente le poids j du neurone i à l'instant t ; η est le taux d'apprentissage, d_i la cible du neurone, a_i la sortie courante du neurone, λ le paramètre de régularisation, m le nombre de données et $\dfrac{\lambda}{m} w_{i,j}$ le terme de régularisation. Le nouvel algorithme est présenté par l'Algorithme 2. La principale particularité de cet algorithme est l'étape de discrétisation des données dont la complexité en temps est de $O(n)$ ou de $O\big(nlog(n)\big)$ [14] selon que la discrétisation se fasse de manière supervisée ou non.

4. Résultats et discussions

Dans notre étude expérimentale nous considérons trois jeux de données provenant des bases d'apprentissage de l'UCI Irvine [24]. Ces jeux de données sont présentés brièvement à la Table 1.

Base	# Données	# Attribut	# Classes
Bank Marketing	45211	17	2
Connectionist Bench	990	10	11
Diabète	768	9	2

Tableau 1. *Jeux de données*

Nous comparons notre algorithme avec l'algorithme de Keramitsoglu et al. et l'algorithme classique des RBF. Différentes techniques de discrétisation ont été utilisées dans l'implémentation de notre algorithme : Sturges, Scott, Freedman-Diaconis (FD), CACC, CAIM. Nous allons également observer systématiquement l'effet des différents paramètres que suit : l'utilisation d'un modèle gaussien multi-varié sur la couche cachée, l'ajout du paramètre de régularisation sur la couche de sortie. Afin d'obtenir des performances de généralisation fiables nous effectuons une validation croisée à dix ensembles (10-fold cross validation).

4.1. Discrétisation des attributs

Nous commençons par vérifier expérimentalement la stabilité et la reproductivité des résultats de notre algorithme quel que soit l'ordre de passage des exemples. Pour cela nous avons utilisé le jeu de données „Bank Marketing“. Nous avons calculé le nombre moyens et l'écart type du nombre de neurones cachés obtenu sur 100 itérations en permuttant à chaque fois l'ordre de passage des exemples avec la méthode de Sturges. Nous avons obtenu 804 neurones en moyenne et un écart type égale à 9 soit 1.12% du nombre moyen de neurones cachés. Ceci permet donc de constater que l'ordre de passage des points n'a pas un grand impact sur le nombre de centres des neurones RBF.

Les performances présentées ici et dans les sous-sections suivantes sont obtenues sur le jeu de données „Bank Marketing“. Nous effectuons une discrétisation des attributs et nous observons l'impact sur les résultats de classification. Les résultats obtenus (taux d'apprentissage, temps moyen d'exécution, nombres de neurones RBF) sont présentés à la Table 2. Ces résultats montrent que la discrétisation améliore les performances dans la plupart des cas. Principalement avec les algorithmes de discrétisation non supervisés, nous observons des taux supérieurs à 86% contrairement à l'algorithme de Keramitsoglu et celui des RBF classique où les taux ne dépassent pas les 84%. De plus les temps d'exécution principalement avec la méthode de Sturges et celle de Freedman-Diaconis sont relativement faible montrant ainsi que le nombre d'ensemble flou considéré ici a permis de générer un nombre relativement petit mais suffisant de neurones sur la couche cachée. Nous avons par exemple 18 neurones sur la couche cachée avec Freedman-Diaconis mais nous atteignons des performances supérieures à l'algorithme de Keramitsoglu dont le nombre de neurones est relativement plus important. Ceci met en évidence une couverture inadéquate de l'espace d'entrée par l'algorithme de Keramitsoglu. Une remarque importante est faite au niveau des méthodes supervisées CACC et CAIM, où malgré un nombre de neurones réduit sur la couche cachée, ces algorithmes présentent des longs

temps d'exécution lié à la discrétisation supervisée des attributs et des taux de classification ne depassant pas les 80%.

Algorithme		Bank Marketing		
		Taux	Temps	# RBF
RBF basé densité	STURGES	**89.11%**	$5.09s$	566
	SCOTT	88.27%	$77.47s$	3068
	FD	86.18%	**0.27s**	**18**
	CACC	79.58%	$64.39s$	10
	CAIM	74.08%	$18.43s$	3
Keramitsoglu et al.		83.98%	$7.52s$	694
RBF classique		77.28%	$4.86s$	20

Tableau 2. *Performances sur le jeu de données Bank Marketing*

4.2. Utilisation du modèle gaussien multi-varié

Nous désirons observer l'impact de l'utilisation d'un modèle gaussien multi-varié pour calculer les sorties des neurones RBF. A cet effet nous présentons à la Table 3, les résultats obtenus en appliquant une discrétisation avec la méthode de Sturges à laquelle nous ajoutons le modèle gaussien multi-varié.

Algorithme		Bank Marketing		
		Taux	Temps	# RBF
RBF basé densité	STURGES	89.11%	$5.09s$	566
	STURGES + GAUSSIEN MULTI-VARIÉ	**91.72%**	**7.70s**	566
Keramitsoglu et al.		83.98%	$7.52s$	694
RBF classique		77.28%	$4.86s$	20

Tableau 3. *Performances sur le jeu de données Bank Marketing avec un modèle gaussien multi-varié*

Nous pouvons observer une légère amélioration due à l'ajout du modèle gaussien multi-varié qui permet de capturer plus de corrélation entre les attributs, avec un temps d'exécution légèrement plus élevé.

4.3. Utilisation du paramètre de régularisation

L'apprentissage des réseaux RBF est principalement axé sur la couche cachée. Cependant l'apprentissage des poids sur la couche de sortie est aussi importante.

Pour cela nous ajoutons un paramètre de régularisation sur la couche de sortie pour réduire l'amplitude des poids des neurones, évitant ainsi les problèmes de surapprentissage. Les résultats obtenus avec la méthode de discrétisation de Sturges à laquelle nous ajoutons ce paramètre de régularisation sont presentés à la Table 4.

Nous observons également une légère amélioration de plus de 2% due à l'ajout du paramètre de régularisation.

4.4. Performances sur les autres jeux de données

Les résultats obtenus (taux d'apprentissage et temps moyen d'exécution) sur les trois jeux de données sont présentés à la Table 5. Ici nous combinons la discrétisation des

Algorithme		Bank Marketing		
		Taux	Temps	# RBF
RBF basé densité	STURGES	89.11%	5.09s	566
	STURGES + RÉGULARISATION	**92.04%**	**6.16s**	566
Keramitsoglu et al.		83.98%	7.52s	694
RBF classique		77.28%	4.86s	20

Tableau 4. *Performances sur le jeu de données Bank Marketing avec un paramètre de régularisation*

attributs, l'utilisation du modèle gaussien multi-varié et l'utilisation du paramètre de régularisation.

Algorithme		Bank		Connectionist		Diabète	
		Taux	Temps	Taux	Temps	Taux	Temps
Densité	STURGES	**93.56%**	7.85s	84.34%	**0.812s**	**71.05%**	**0.058s**
	SCOTT	91.27%	78.02s	95.40%	1.33s	58.59%	0.314s
	FD	89.32%	**2.10s**	**97.02%**	1.861s	67.44%	0.102s
	CACC	80.83%	66s	70.75%	158.2s	62.84%	29.68s
	CAIM	73.04%	20.52s	43.63%	74.40s	65.10%	2.453s
Keramitsoglu et al.		83.98%	7.52s	84.39%	0.893s	53.38%	0.102s
RBF classique		77.28%	4.86s	58.83%	0.492s	56.77%	0.364s

Tableau 5. *Taux d'apprentissage et temps d'exécution des algorithmes*

Ces résultats sur d'autres jeux de données montrent également que le nouvel algorithme est assez stable (en utilisant une des methodes, Sturges, Scott ou FD) comparé a l'algorithme de Keramitsoglu dont les taux de classification bien que supérieurs à ceux des RBF classique en général présentent quelques signes d'instabilité comme c'est le cas avec les données du *Diabète* où la précision de Keramitsoglu et al. (53.38%) est inférieur à celle de l'algorithme RBF classique (56.77%).

5. Conclusion

Nous avions pour objectif de proposer un nouvel algorithme de construction des réseaux RBF pour la classification qui exploite au mieux la densité des exemples d'apprentissage. Partant du constat que la distribution des données d'apprentissage sur chaque axe des caractéristiques joue un rôle important, nous avons pu grâce aux techniques de discrétisation mais aussi de régularisation développer un algorithme dont les caractéristiques principales sont la stabilité et la précision par rapport à l'algorithme de Keramitsoglu et al. dont il est dérivé

Les résultats que nous avons obtenus nous ont permis de constater que parmi les méthodes de discrétisation utilisées, il n'existe pas une qui surpasse réellement les autres, ceci est un résultat un peu attendu car le le problème de déterminer la solution optimale des points de coupe pour discrétiser les données est NP-difficile ; cependant dans la littérature des méthodes de discrétisation sont parfois développées par rapport au traitement futur a réaliser, pour cela il serait intéressant en fonction des données à classifier de dériver à partir de l'algorithme proposé dans cet article de nouvelles versions qui implémentent des méthodes de discrétisation mieux adaptées. De plus une analyse théorique

des propriétés du nouvel algorithme doit être faite afin d'identifier de manière formelle les conditions de son bon fonctionnement.

6. Bibliographie

[1] KERAMITSOGLOU, SARIMVEIS, H. W., KIRANOUDIS, C. T., SIFAKIS, N., « Radial basis function neural networks classification using very high spatial resolution satellite imagery : an application to the habitat area of Lake Kerkini (Greece) », *International Journal of Remote Sensing*, vol. 26, n° 9, pages 1861-1880, 2005.

[2] RUMELHART, D.E., G.E. HINTON, R.J. WILLIAMS, « Learning internal representations by error propagation », *D.E. Rumelhart and J.L. McClelland, Cambridge, MA : The MIT Press. Parallel Distributed Processing : Explorations in the Microstructure of Cognition*, vol. 1, pages 318-362, 1986.

[3] POWEL, M.J.D., « Radial basis functions for multivariable interpolation : A review », *IMA Conference on Algorithms for the Approximation of Functions and Data,RCMS, shrivenham, England*, pages 143-167 1985.

[4] SARIMVEIS, H., ALEXANDRIDIS, A., TSEKOURAS, G., BAFAS, G., « A fast and efficient algorithm for training radial basis function neural networks based on a fuzzy partition of the input space », *Industrial and Engineering Chemistry Research*, vol. 41, pages 751-759, 2002.

[5] MADHU, G., RAJINIKANTH, T.V., GOVARDHAN, A., « Improve the classifier accuracy for continuous attributes in biomedical datasets using a new discretization method », *2nd International Conference on Information Technology and Quantitative Management, ITQM*, vol. 31, pages 671-679, 2014.

[6] LIZHONG WU, JOHN MOODY, « A Smoothing Regularizer for Feedforward and Recurrent Neural Networks », *Neural Computation*, vol. 8, n° 3, pages 463-491, 1996.

[7] TAO XIAOLI, HOWARD, E. M., « Classification of multi-spectral satellite image data using improved NRBF neural networks », *University of Massachusetts Dartmouth, Dartmouth MA 02747*, 1999.

[8] SING, J.K., BASU, D.K., NASIPURI, M., KUNDU, M., « Improved k-means algorithm in the design of RBF neural networks », *TENCON 2003. Conference on Convergent Technologies for the Asia-Pacific Region*, vol. 2, pages 841 - 845, 2003.

[9] QINGYU XIONG AND HIRASAWA, K, JINGLU HU, MURATA, J., « Growing RBF structures using self-organizing maps », *Robot and Human Interactive Communication, 2000. RO-MAN 2000. Proceedings. 9th IEEE International Workshop on*, pages 107 - 111, 2000.

[10] LEONARD, J.A., KRAMER, M.A., « Radial basis function networks for classifying process faults », *IEEE Control Systems*, vol. 11, pages 31-38, 1991.

[11] B.WIDROW, M.E.HOFF, « Adaptive switching circuits », *Wescon Convention Record*, pages 96-104, 1960.

[12] YAMASHITA, K., CHAKRABORTY, G., MABUCHI, H., MATSUHARA, M., « An Efficient Method to Set RBF Network Paramters Based on SOM Training », *Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on*, pages 426 - 431, 2010.

[13] CIOS, K.J, KURGAN, L.A., « CLIP : Hybrid Inductive Machine Learning Algorithms that Generates Inequality Rules », *Information Science*, vol. 163, pages 37-83, 2004.

[14] TSAI, C. J., LEE, C. I., YANG, W. P, « A Discretization Algorithm based on Class-Attribute Contingency Coefficient », *Information Sciences*, vol. 178, pages 714-731, 2008.

[15] KURGAN, L. A., CIOS, K. J., « CAIM Discretization Algorithm », *IEEE Transactions on knowledge and data engineering*, vol. 16, pages 145-153, 2004.

[16] FAYYAD, U. M., IRANI, K. B., « Multi-interval discretization of continuous-valued attributes for classification learning », *Pattern Recognition Letters*, vol. 24, pages 895-902, 2003.

[17] AMITAVA ROY, SANKAR, K.PAL, « Fuzzy Discretization of Feature Space for a Rough Set Classifier », *Artificial intelligence*, vol. 13, pages 1022-1027, 1993.

[18] YING YANG, GEOFFREY, I. WEBB, XINDONG WU, « Discretization Methods », *Data Mining and Knowledge Discovery Handbook Second Edition, O. Maimon, L. Rokach, Eds*, pages 101-116, 2010.

[19] STURGES, H. A., « The choice of a class interval », *J. American Statistical Association*, pages 65-66, 1926.

[20] SCOTT, DAVID W., « On optimal and data-based histograms », *Biometrika*, vol. 66, n° 3, pages 605-610, 1979.

[21] FREEDMAN, DAVID, DIACONIS, P., « On the histogram as a density estimator : L2 theory », *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, vol. 57, n° 4, pages 453-476, 1981.

[22] GONZALEZ-ABRIL, L., CUBEROS, F. J., VELASCO, F., ORTEGA, J. A., « Ameva : An autonomous discretization algorithm », *Expert Systems with Applications*, vol. 36, pages 5327 ?5332, 2009.

[23] GUT, ALLAN, « An Intermediate Course in Probability », *Springer, ISBN 9781441901613 (Chapter 5)*, 2009.

[24] BLAKE, C.L., MERZ, C.J., « UCI Repository of machine learning databases », *http ://www.ics.uci.edu/ mlearn/MLRepository.html*, 1998.